Licenciatura em Engenharia Informática e de Computadores

Relatório Final

Projeto
de
Laboratório de Informática e Computadores
2017 / 2018 verão

Autores:
Tiago Pereira 43592
Luís Guerra 43755
Daniel Carmelino 43574

1 Keyboard Reader

O módulo *Keyboard Reader* implementado é constituído por dois blocos principais: *i*) o descodificador de teclado (*Key Decode*); e *ii*) o bloco de armazenamento e de entrega ao consumidor (designado por *Key Buffer*), conforme ilustrado na Figura 1. Neste caso o módulo de controlo, implementado em *software*, é a entidade consumidora.

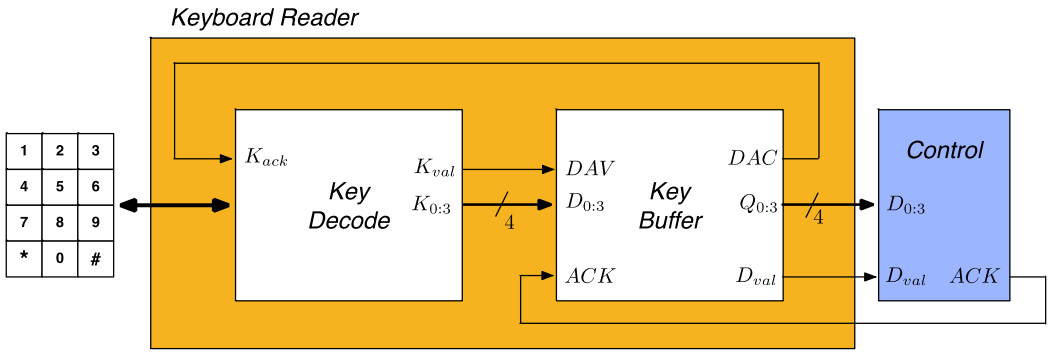
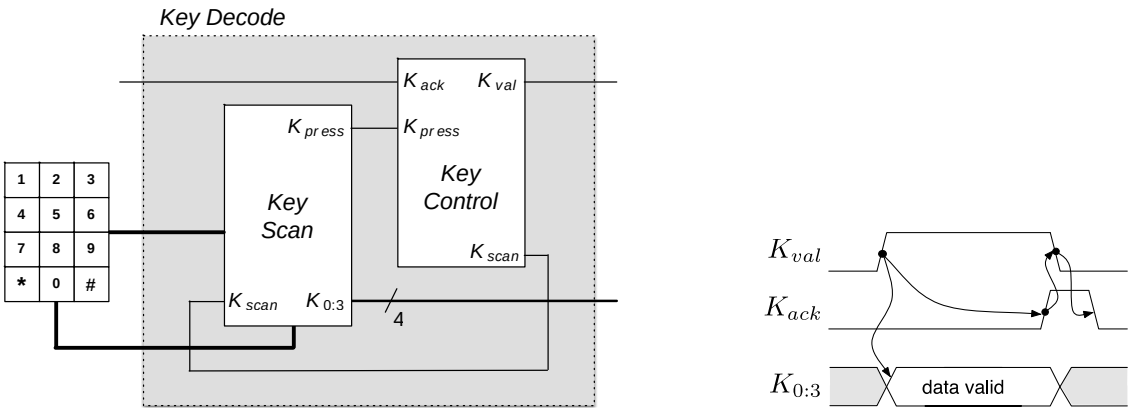


Figura 1 – Diagrama de blocos do módulo *Keyboard Reader*

1.1 Key Decode

O bloco *Key Decode* implementa um descodificador de um teclado matricial 4x3 por *hardware*, sendo constituído por três sub-blocos: *i)* um teclado matricial de 4x3; *ii)* o bloco *Key Scan*, responsável pelo varrimento do teclado; e *iii)* o bloco *Key Control*, que realiza o controlo do varrimento e o controlo de fluxo, conforme o diagrama de blocos representado na Figura 2a.

O controlo de fluxo de saída do bloco *Key Decode* (para o módulo *Key Buffer*), define que o sinal K_{val} é ativado quando é detetada a pressão de uma tecla, sendo também disponibilizado o código dessa tecla no barramento $K_{0:3}$. Apenas é iniciado um novo ciclo de varrimento ao teclado quando o sinal K_{ack} for ativado e a tecla premida for libertada. O diagrama temporal do controlo de fluxo está representado na Figura 2b.



a) Diagrama de blocos b) Diagrama temporal

Figura 2 – Bloco *Key Decode*

O bloco *Key Scan* foi implementado de acordo com o diagrama de blocos representado na Figura 3.

Para implementar o bloco Key Scan optou-se por recorrer ao esquema da versão 1 presente no enunciado do trabalho. Após uma extensa analise das restantes hipóteses elegemos a versão 1, apesar da versão 3 ser a mais económica em termos de uso de impulsos de relógio. Achou-se que por motivos de previsibilidade e fiabilidade do sistema a escolha deveria recair na 1ªopção. Deste modo escolheu-se robustez e conveniência ao invés de instabilidade e poupança.

O bloco *Key Control* foi implementado pela máquina de estados representada em *ASM-chart* na Figura 4.

Para o desenvolvimento do controlo da máquina de estados do *Key Control* achou-se por bem utilizar apenas três estados. O primeiro estado (00) vai ativar sempre a saída *KSCAN*. A passagem do primeiro para o segundo (01) é regulada por *KPRESS*, caso a mesma não se encontre ativa, o programa mantém-se no estado inicial. O segundo estado (01) ativará a saída *KVAL*. A condição que determina se o programa passa do segundo estado para o terceiro (10) é o valor booleano *KACK*. À semelhança da primeira seleção, caso esta não esteja no nível logico'1'o programa mantém o estado corrente. Por fim o terceiro estado. Este, ao contrário dos restantes, não ativa qualquer saída. A permanência neste estado depende de duas variáveis: *KACK* E *KPRESS*. Caso alguma destas variáveis de entrada esteja com o nível lógico '1' o estado não se altera. Por último, a única combinação que coloca a maquina de estados no estado inicial é *KACK* e *KPRESS* com valor lógico 0.

A descrição hardware do bloco *Key Decode* em CUPL encontra-se no Anexo A.

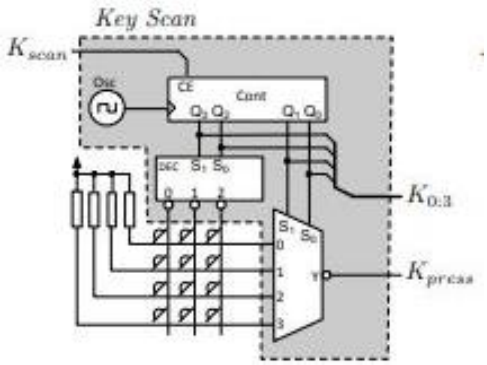


Figura 3 - Diagrama de blocos do bloco *Key Scan*

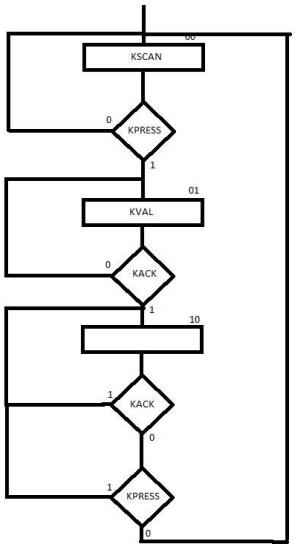
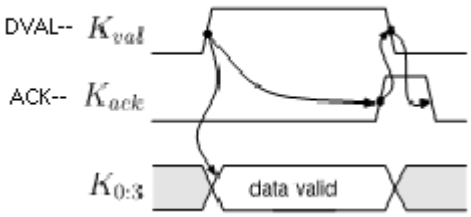


Figura 4 – Máquina de estados do bloco *Key Control*

1.2 Key Buffer

O bloco *Key Buffer* implementa uma estrutura de armazenamento de dados, com capacidade para armazenar uma palavra de quatro bits. A escrita de dados no bloco *Key Buffer*, cujo diagrama de blocos é apresentado na Figura 5, inicia-se com a ativação do sinal *DAV* (*Data Available*) pelo sistema produtor, neste caso pelo bloco *Key Decode*, indicando que tem dados para serem armazenados. Logo que tem disponibilidade para armazenar informação, o bloco *Key Buffer* regista os dados $D_{0:3}$ em memória. Concluída a escrita em memória, ativa o sinal *DAC* (*Data Accepted*) para informar o sistema produtor que os dados foram aceites. O sistema produtor mantém o sinal *DAV* ativo até que o sinal *DAC* seja ativado. O bloco *Key Buffer* só desativa o sinal *DAC* após o sinal *DAV* ter sido desativado. A implementação do bloco *Key Buffer* é baseada numa máquina de estados de controlo (*Key Buffer Control*) e num registo, designado por *Output Register*.
O sub-bloco *Key Buffer Control* do bloco *Key Buffer* também é responsável pela interação com o sistema consumidor, neste caso o módulo *Control*. Quando pretende ler dados do bloco *Key Buffer*, o módulo *Control* aguarda que o sinal D_{val} fique ativo, recolhe os dados e ativa o sinal *ACK* para indicar que estes já foram consumidos. Logo que o sinal *ACK* fique ativo, o módulo *Key Buffer Control* invalida os dados baixando o sinal D_{val}. Para que uma nova palavra possa ser armazenada é necessário que o módulo *Control* desative o sinal *ACK*.
A máquina de estados do *Key Buffer Control* é representada em *ASM-chart* na Figura 6.

Para o desenvolvimento do controlo da máquina de estados do *KeyBuffer Control* foram utilizados quatro estados. O primeiro estado (00) apenas espera a ativação de *DAV* passando assim para o estado seguinte, no segundo estado (01) apenas são ativadas *Wreg* e *DAC* (quando *DAC* fica ativo sabemos que o *DAV* fica com o valor lógico '0' dando-se a passagem para o estado seguinte (10) em que é ativo *DVAL* e é esperada a ativação de *ACK* para haver passagem para o ultimo estado (11) que espera pela desativação de ACK para retornar ao primeiro estado. Baseamo-nos no diagrama temporal a seguir apresentado para realizar esta máquina de estados:



A descrição hardware do bloco *Key Buffer* em CUPL/VHDL encontra-se no Anexo B.

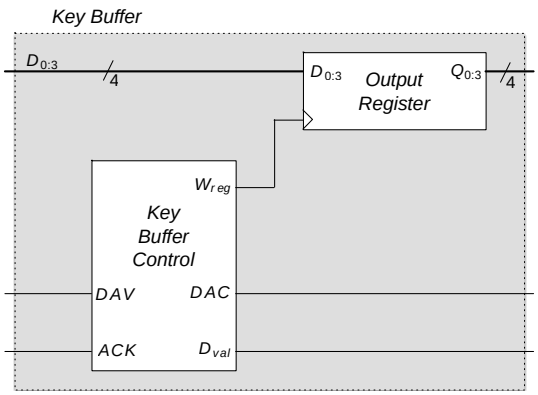


Figura 5 – Diagrama de blocos do bloco *Key Buffer*

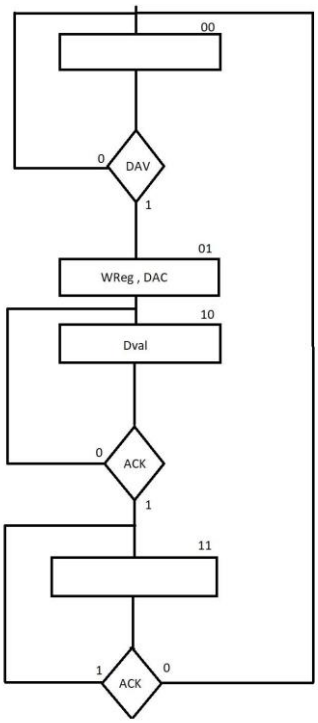


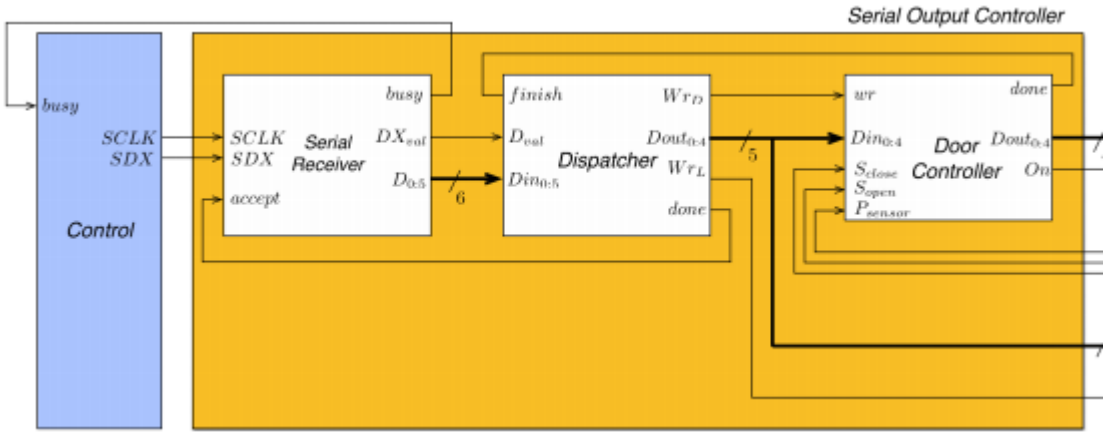
Figura 6 – Máquina de estados do bloco *Key Buffer Control*

Com base nas descrições dos blocos *Key Decode* e *Key Buffer* implementou-se o módulo *Keyboard Reader* de acordo com o esquema elétrico representado no Anexo C.

Para o módulo Keyboard Reader houve a necessidade de usar resistências para a entrada dos dados do teclado sendo que optou-se por resistências (todas com o mesmo valor para não haver variações na eficácia de resposta) com um valor de 330Ω pois após uma leitura das especificações da PAL750C verificamos que a corrente de pinos de output seria de 16mA e utilizando a lei de OHM (5V/16mA = 312,5 ohm) sendo que a resistência mínima para o sistema seria a de 330 ohm. Após uma verificação de qual o clock mais adequado optou-se pelo de 10 KHz pois e o que se adequa melhor a nossa montagem visto que o de 1KHz dava uma resposta demasiado lenta tendo também a atenção á possível existência de *bounce* que é anulada com o aumento da frequência.

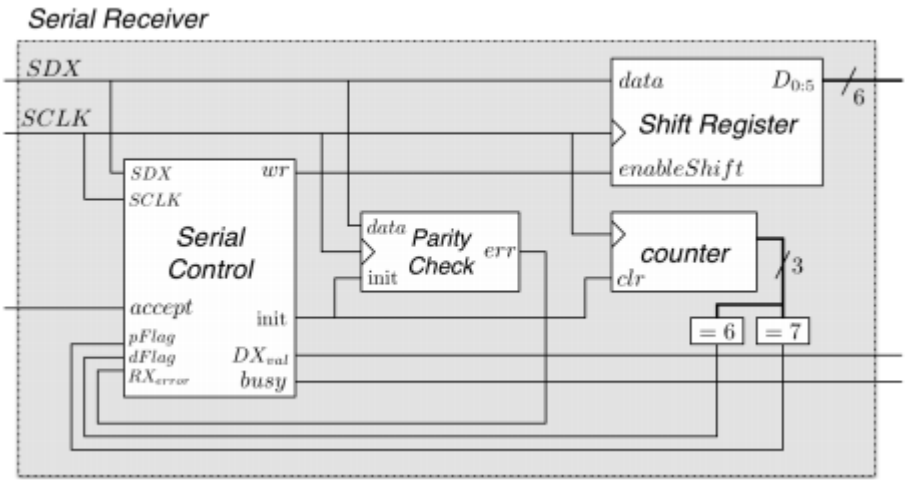
2 Serial Output Controller

O módulo Serial Output Controller (SOC) implementa a interface com o LCD e com o mecanismo da porta, fazendo a receção em série da informação enviada pelo módulo de controlo e entregando-a posteriormente ao destinatário, conforme representado na Figura.



2.1 Serial Receiver

O bloco Serial Receiver do módulo SOC é constituído por quatro blocos principais: i) um bloco de controlo; ii) um bloco conversor série paralelo; iii) um contador de bits recebidos; e iv) um bloco de validação de paridade, designados por Serial Control, Shift Register, Counter e Parity Check respetivamente. O bloco Serial Receiver deverá ser implementado com base no diagrama de blocos apresentado na Figura.



Dividiu-se o bloco Serial Control em duas partes para melhor eficiência e compreensão. Em primeiro lugar o ASM da máquina de estados responsável pela ativação do sinal *Start*.

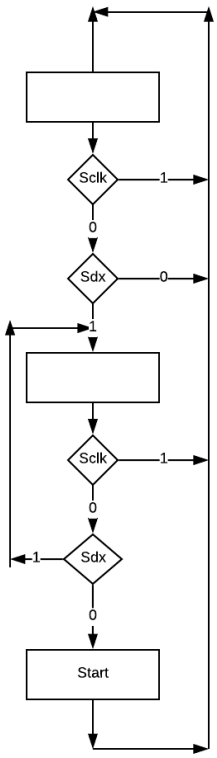


Figura 7 - Máquina de estados do bloco Start

Em segundo lugar o ASM da máquina de estados responsável pela ativação dos restantes sinais.

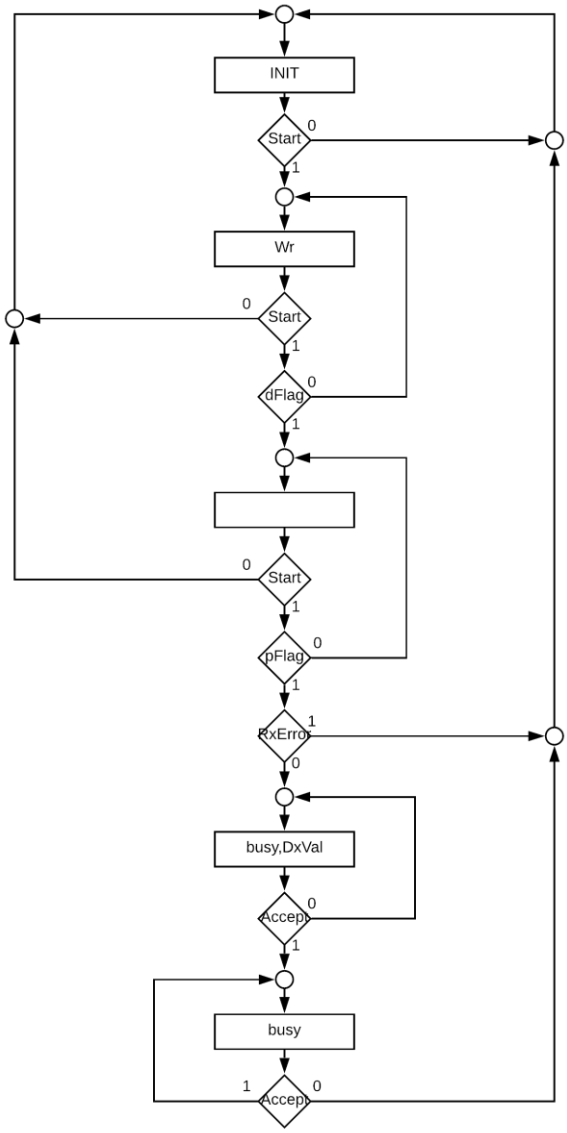


Figura 8 - Máquina de estados
do bloco Control

2.2 Dispatcher

O bloco Dispatcher é responsável pela entrega das tramas válidas recebidas pelo bloco Serial Receiver ao LCD e ao Door Controller, através da ativação do sinal WrL e WrD. A receção de uma trama válida é sinalizada pela ativação do sinal Dval. O processamento das tramas recebidas pelo SOC, para o LCD ou para o Door Controller, deverá respeitar os comandos definidos pelo fabricante de cada periférico, devendo libertar o canal de receção série logo que seja possível.

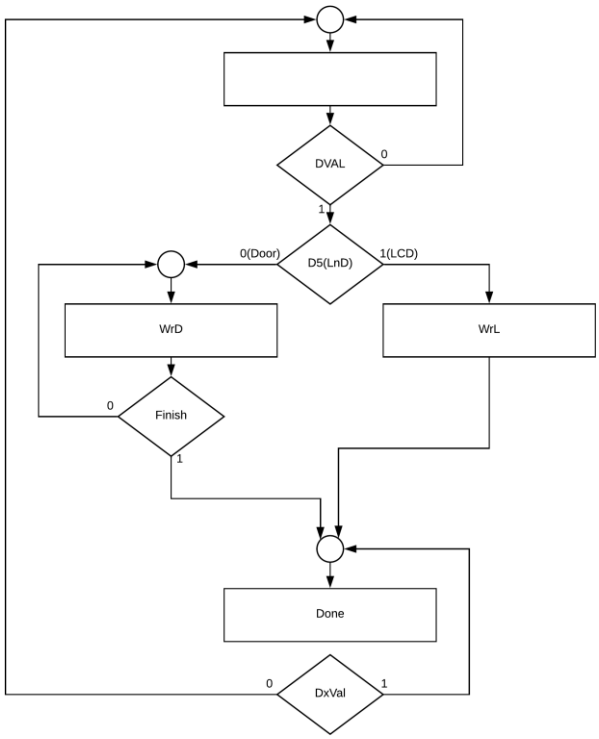


Figura 9 - Máquina de estados do bloco Dispatcher

2.3 Door Controller

O bloco Door Controller após ter recebido um comando válido entregue pelo Dispatcher, deverá proceder à atuação deste no mecanismo da porta. Se o comando recebido for de abertura da porta, o Door Controller deverá colocar o sinal *OnnOff* e o sinal *OpennClose* no valor lógico '1', até o sensor de porta aberta (FCopen) ficar ativo. No entanto se o comando for de fecho, o Door Controller deverá ativar o sinal *OnnOff* e colocar o sinal *OnnOff* no valor lógico '0', até o sensor de porta fechada (FCclose) ficar ativo. Se durante o fecho for detetada uma pessoa na zona da porta, através do sensor de presença (Pdetect), o sistema deverá interromper o fecho reabrindo a porta. Após a interrupção do fecho da porta o bloco Door Controller deverá permitir de forma automática, ou seja, sem necessidade de envio de um novo comando, o encerramento da porta e o finalizar do comando de fecho. Após concluir qualquer um dos comandos o Door Controller sinaliza o Dispatcher que está pronto para processar um novo comando através da ativação do sinal done.

De referir as igualdades de variáveis dentro do programa pois os módulos Dispatcher e Door Controller estão presentes na mesma PAL não necessitando de pinos de entrada ou saída para os mesmos. De saída dessa PAL estará a variável *done* que é a entrada *accept* do *SerialReceiver* que se encontra numa PAL diferente onde nessa, encaminha a variável *busy* para o modúlo Control.

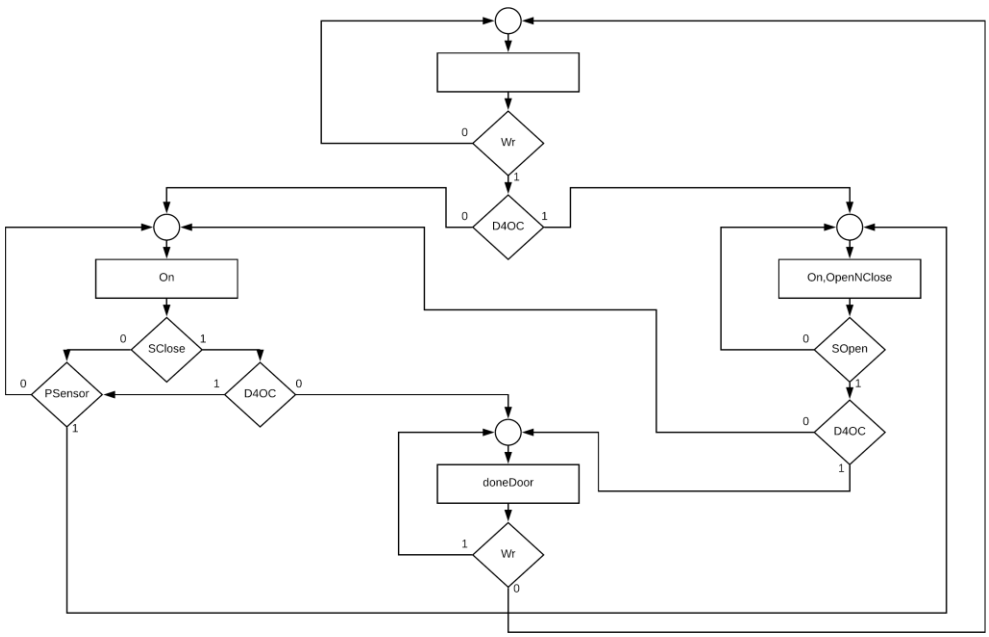


Figura 10 - Máquina de estados do bloco Door Controller

3 Control

Implementou-se o módulo *Control* em *software*, recorrendo a linguagem Java e seguindo a arquitetura lógica apresentada na Figura 11.

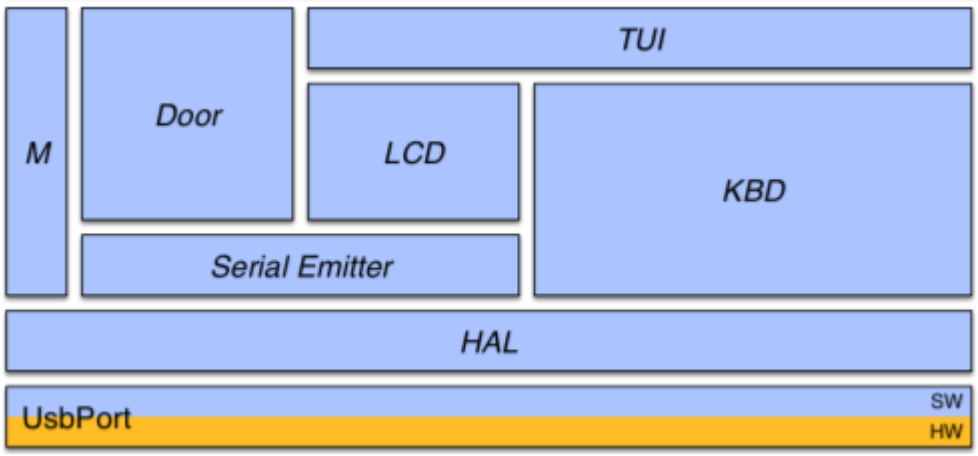


Figura 11 – Diagrama lógico do módulo *Control*

3.1 Classe *HAL*

A classe HAL é essencial para a correta interação entre o *UsbPort* e as restantes classes do projeto. *HAL* ou *Hardware Abstraction Layer* serve para, tal como o nome indica, criar uma camada de abstração para assim todas as outras classes não terem que recorrer ao *UsbPort* para recolha ou impressão de dados. Deste modo, a classe *HAL* funciona como um intermediário ou agente facilitador de todas as outras classes que têm interação direta com hardware. Por último, mas não menos importante, caso haja alteração de hardware apenas esta classe precisa de ser repensada visto que todas as outras não têm ligações diretas com o *UsbPort*.

Nota: Adicionou-se dois métodos auxiliares, *in() e out()*, que recorrem ao método *in() e out()* , respetivamente, da classe *UsbPort* presente no package isel.leic negando as variáveis do *UsbPort*, tendo com finalidade a melhor leitura e compreensão do programa.

3.2 Classe *KBD*

A classe *KBD* serve para se obter em software o código das teclas recebidas como input ao *UsbPort* que através da *HAL* chegam ao *software*. Portanto esta classe irá interagir diretamente com a classe *HAL* de maneira em que definimos a máscara *KEY* para irmos buscar os bits pretendidos (coincidentes ao código da tecla) ao *UsbPort* através sempre da *HAL* e dos seus respetivos métodos. Para o funcionamento correto deste procedimento é respeitada uma sequência de acontecimentos que será verificarmos se existe bit (isBit), ou seja tecla premida, com a máscara *DVAL*, lermos os bits correspondentes à tecla através da máscara *KEY* e guardá-los, ativar *ACK* e enquanto estivermos com a tecla premida ficará preso o programa e após a libertação da tecla o *ACK* é desativado dando limpando todos os bits, fazendo-os retornar ao valor lógico ‘0’. De referir que este funcionamento está suportado num "temporizador" em que durante o tempo definido se pressionada a tecla retorna-a, se não pressionada retorna *NONE*.

3.3 Classe LCD

A classe LCD permite nos numa primeira instancia estabelecer contacto com uma nova peça de hardware, dispositivo o qual nos vai permitir visualizar inputs dados pelo utilizador, com a interação do UsbPort. A classe LCD propriamente dita tem como propósito escrever no display do LCD usando uma interface de 4 bits. O display terá uma dimensão de 2 linhas e 16 colunas para podermos inserir carateres. Esta classe vai maioritariamente assentar em 3 métodos que necessitam da consulto e entendimento do dispositivo em si que serão o método de inicialização (*init()*), o método que escreve um nibble de comando ou dados no LDC (*writteNibble(boolean rs, int data)* e por fim o método que escreve só um byte de comando ou dados no LCD (*writeByte(boolean rs, int data)* . De referir as variáveis envolvidas neste processo: RS (Register Select) que indica se são dados ou comando; R/W que estará sempre a zero; E (enable) e ainda bits de data.

Começando pelo método de inicialização do dispositivo e ao consultar o *Quick Refernce Guide* disponibilizado pelos docentes verificamos que na secção de inicialização usando uma interface de 4 bit (figura 8) existe um diagrama temporal que terá de ser respeitado para que a inicialização seja bem concluída. Este processo envolve a codificação de 6 bits: RS R/W DB7 DB6 DB5 e DB4.

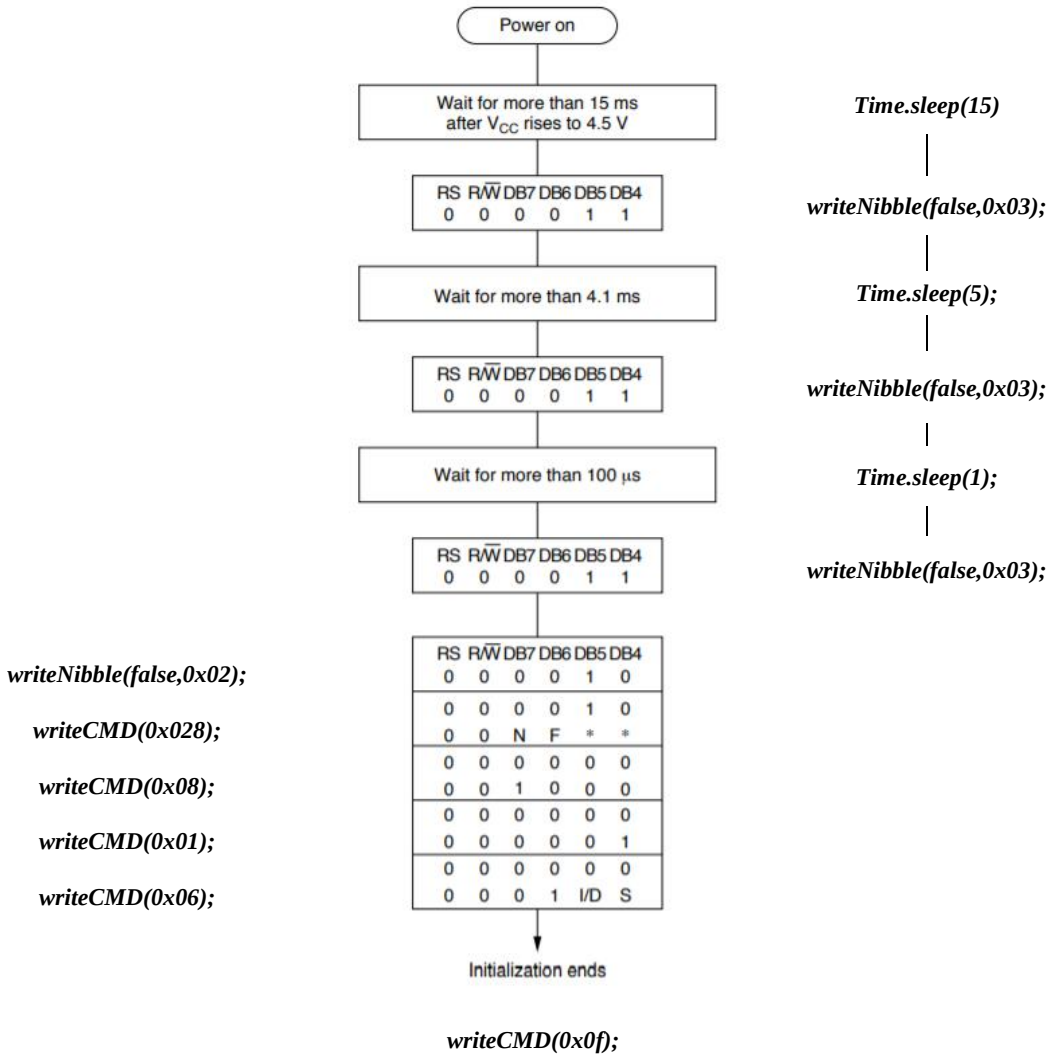


Figura 12 - Diagrama de inicialização do LCD

De referir que a espera de tempos é realizada pela função *Time.sleep()* e que o seu tempo mínimo aceitável como parâmetro é de 1 ms portanto todas as esperas de tempo abaixo dessa unidade terão de ser arredondados para 1ms. Também como a função so aceita números inteiros os tempos com casas decimais terão de ser arredondados para o inteiro acima.

Na instrução writeCMD(0x028) são apresentadas duas variáveis N e F. N é um bit que se refere a : N = 1: 2 linhas, N = 0: 1 linha(a escolha recai em por o bit N a 1) ; F é um bit que representa se o utilizador pretende o display a : F = 1: 5 x10 dots ou F = 0: 5 x 8 dots (a escolha recai em F = 0).

Na última instrução deparou se com mais 2 variáveis booleanas I/D e S que representam respetivamente se o utilizador quer incrementar ou decrementar e se queremos que acompanhe o shift do display. A escolha recai sobre queremos incrementar (I/D =1) e não acompanhar (S=0).

Passamos agora para o método writeNibble que será a base de todos os métodos de escrita no display, este método terá de respeitar um diagrama temporal que ira envolver os bits RS, W/R, E e DATA. O método consiste em jogar com esses bits de modo a que em primeira instância coloque RS =0, depois E=1, DATA = 1 e por fim baixar o E para 0 outra vez como podemos verificar na figura abaixo. Respeitou-se também a espera de tempos todas impostas 1 ms.

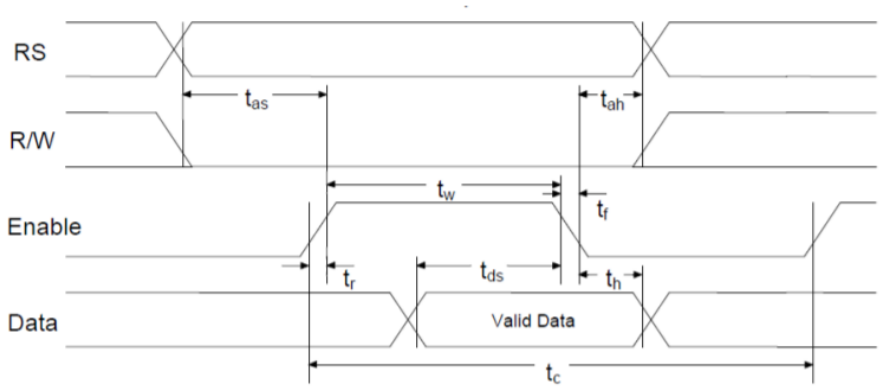


Figura 13 - Protocolo de transferência de dados do LCD

Como este método trabalha em conjunto com métodos da classe HAL teve se de criar constantes com as máscaras que permitiam localizar no Usbport os bits de RS, E e os dados (DATA).
Por fim o método writeByte que escreve só 1 byte, como 1 byte são 8 bits e a nossa interface so permite enviar tramas a 4 bits teremos de usar duas vezes o método writeNibble para receber os 2 conjuntos de 4 bits compostos do byte.
Estes foram os 3 métodos base para todos os outros de escrita e comando:

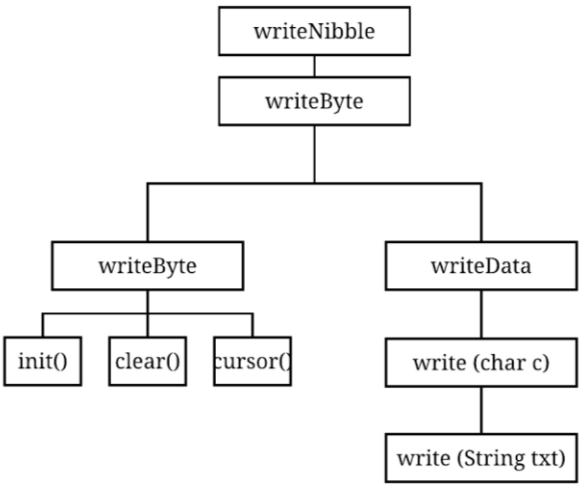


Figura 14 - Hierarquia de métodos da classe LCD

3.4 Classe *SerialEmitter*

A classe SeriaEmitter é a classe que entrará em contacto direto com o modelo de hardware SOC (serial output controller), através mais precisamente do seu módulo SerialReceiver que irá receber as tramas enviadas por este pedaço de software através dos sinais SCLK e SDX que dirá o tipo de dados bem como o seu destino(DOOR/LCD).
Esta classe terá de responder única e simplesmente a um protocolo temporal que será o seguinte:

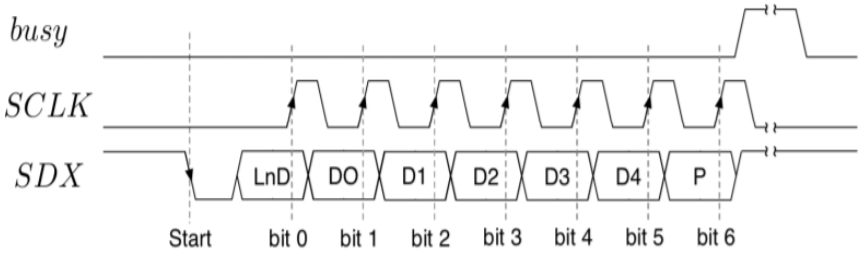


Figura 15 - Protocolo de comunicação com o módulo Serial Output Controller

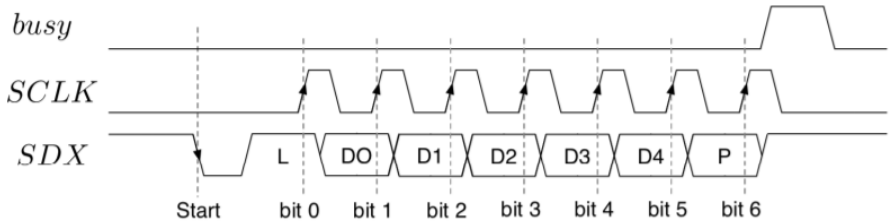


Figura 16 - Trama para o LCD

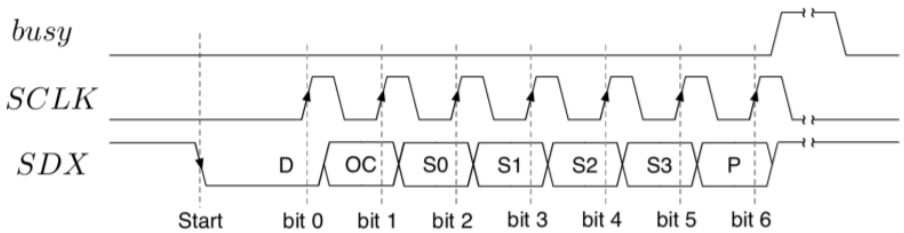


Figura 17 - Trama para o mecanismo da porta (Door)

Como se pode verificar na primeira figura está representada a ordenação dos 7 bits enviados e o seu significado respetivamente. No modelo geral verificou-se que o primeiro bit LnD identifica o destinatário dos restantes bits (1=LCD, 0=Door) e o último bit representa a paridade da trama que serve para detetar erros na transmissão.
Após a escolha do seu destinatário os bits terão significados diferentes conforme claro o seu destinatário. No caso em que a trama se destina ao LCD o primeiro bit será de informação e indica se a mensagem é de controlo ou dados(RS) sendo os restantes bits de dados e de paridade. No caso em que a trama se destina ao mecanismo da porta temos que o primeiro bit terá a informação que nos dirá se queremos abrir ou fechar a porta e os restantes sendo de velocidade do motor e paridade.

Explicada agora a ordenação e significado dos bits que a trama contém passa -se agora á realização da classe propriamente dita. Olhando para o diagrama geral temporal apresentado verificou-se a obrigação de respeitar 2 aspetos fundamentais: a condição de start criando um método á parte (start()) e o estado de SCLK aquando da transmissão de SDX.
Primeiramente a condição de start será colocar o SCLK a 0 e fazer a transição do SDX de 1 para 0.

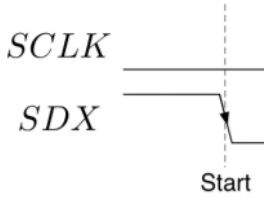


Figura 18 - Condição de Start

Após implementado este protocolo de iniciação verificou-se que a transmissão(transição) de bits ocorre sempre
quando do sinal SCLK esta a 1 sendo assim os clocks estão “alternados” tendo então que assegurar isso no desenvolvimento do programa mais precisamente na parte do envio de bits.

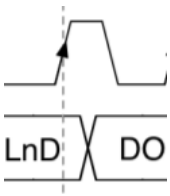


Figura 19 - Transição de bits

Por fim realizou-se o método isBusy() que dirá se a trama esta acabada, este método irá comunicar com o método readBits da classe HAL que irá ler o bit mostrado no usbPort e que irá ser recebido então pelo software dizendo se esta busy ou não (1= busy).

3.5 Classe Door

A classe Door será implementada com a finalidade de controlar o mecanismo da porta enviando os respetivos comandos de abertura, fecho e conclusão da abertura ou fecho de porta.

Esta classe comunica com a classe SerialEmitter na medida em que nos métodos open e close envia o comando para o DoorMechanism e com uma determinada velocidade sendo esse o parâmetro dos métodos. De referir que o que irá diferenciar os métodos será a informação do bit OC.

Para o método open shiftamos os bits para a esquerda 1x e concatenamos com 1 pois esse bit deverá estar a 1 enquanto que no close deverá estar a 0.

Para o último método isFinshed() vai interagir com o método isBusy() da classe SerialEmitter pois só com essa informação saberemos que a porta estará totalmente aberta ou fechada.

3.6 Classe TUI

Esta classe tem como objetivo apresentar uma camada de abstração entre a App e os métodos que comunicam com o hardware como por exemplo a classe LCD e a classe KBD, por isso se for necessária por alguma razão a mudança de hardware a classe App será salva guardada.

Portanto nesta classe serão implementados métodos de suporte á classe App que fazem a transição de outras classes para TUI e depois para a App.

3.7 Classe File Access

Esta classe tem como seu foco o acesso ao ficheiro “USERS.txt” e extração de cada uma das suas linhas o utilizador que contêm as suas informações. Para esse fim leu-se cada linha do ficheiro e visto que cada atributo do utilizador está separado pelo carater “;”, utilizou-se a função split da classe String que recebe a linha do ficheiro e sempre que encontra esse carater retorna a string até lá passando a para inteiro nos casos de identificação, password e tempo acumulado. E dentro desse ciclo repete se a criação de cada objeto da classe User com esses parâmetros como também a sua inserção na estrutura de dados criada.

3.8 Classe Users/User

A classe Users trata tudo o que tem a ver com o utilizador e as suas propriedades. Primeiramente comunica com a classe FileAccess para carregar todos os utilizadores do ficheiro para o array de objetos da classe User criada nesta classe, ou seja, será uma classe interna. Ainda na classe Users foram implementados alguns métodos uteis e/ou possivelmente uteis para a realização de outras classes como procurar um utilizador e obter a sua password.

A classe User (interna á classe Users) tem como objetivo a instanciação e criação dos objetos que remetem para um utilizador. Nesta classe instanciam-se a pass, a identificação do utilizador, o seu nome, o tempo acumulado, uma variável para dizer se o utilizador está a entrar ou a sair e também a sua hora de entrada. Nesta classe é criado o construtor para transportar os utilizadores para a estrutura de dados utilizada e para a função de criação de utilizador em modo de manutenção. Por fim foram criados setters e getters de acordo com as necessidades de obtenção dos seus atributos.

3.9 Classe M

Esta classe denominada de M (Manutenção) tem um simples objetivo realizado em apenas um método que é verificar se o utilizador pressionou o botão M que remete á manutenção, ou seja, verifica o estado do botão (lendo a partir do UsbPort) e retorna true ou false. Se retornar true então terá de ser iniciado o processo de manutenção do sistema em qualquer estado que a máquina esteja, ou seja, o botão poderá ser pressionado a qualquer instância.

3.10 Classe Maintenance

Esta classe tem por objetivo iniciar e controlar o processo de manutenção após ter sido pressionado o botão M. Processo que consiste no seguinte: Após o botão ter sido pressionado a mensagem de manutenção aparece escrita na linha de comandos. O utilizador terá á sua disposição 4 comandos neste modo: NEW, DEL, LST e OFF. NEW criará um novo objeto, ou seja, um novo utilizador. DEL apagará um utilizador pretendido. LST mostrará a lista de utilizadores que estão dentro do sistema (InNOut=1) e OFF acabará o programa.

3.11 Classe Log

Esta classe tem por seu objetivo o acesso ao ficheiro “LOG.TXT” e registar as entradas e saídas dos utilizadores, ou seja, terá de ser reescrito sempre que haja uma entrada ou saída de utilizador. Em uma linha desse ficheiro esta a informação da data de entrada/saída, a hora exata, o tempo acumulado, indicador de entrada ou saída, identificação e nome do utilizador.

O indicador de entrada ou saída será: “->” – entrada e “<- “– saída. Esta decisão será suportada pelo indicador booleano que esta presente na classe User chamado de InNOut que estará a true se o utilizador estiver fora e for entrar e a false se estiver dentro e for sair.

3.12 Classe App

Esta classe é a "classe mãe" deste projeto pois é aqui onde as outras classes e seus respetivos métodos serão aplicados de forma a chegar a um objetivo. Esta classe é o motor de toda a máquina, ou seja, é onde a cronologia dos acontecimentos terá de ser representada bem como a transição e demonstração gráfica de valores e dados no LCD.

O funcionamento da máquina será o seguinte: começará por mostrar graficamente a data do dia bem como a hora e pedirá a identificação do utilizador e enquanto não for pelo menos tentada o display será sempre o mesmo. Após a inserção de uma identificação valida irá ser requisitada a password e então comparou-se a password escrita pelo utilizador com a password do utilizador que foi procurado através da inserção da identificação. Se coincidir dar se á a entrada do utilizador, de referir que por escolha do grupo o utilizador estará sempre a entrar se a máquina for ligada, ou seja, a máquina estará sempre ligada desde que haja utilizadores dentro do sistema.

Após a validação do utilizador aparecerá no display o dia da semana formatado e a hora atual. De seguida o aviso de abertura e fecho de porta bem como o mecanismo visual de abertura e fecho de porta (Door) respeitando que se esta a fechar verificar o sensor de pessoa.

Voltamos então ao display inicial visto que já se deu a entrada do utilizador. De referir que no ficheiro LOG.txt já esta neste ponto indicada a entrada do utilizador. Estando então outra vez no inicio haverá 2 situações: ou são inseridos os dados de alguém que se encontra no interior ou de alguém que ainda não entrou e voltará a acontecer o mesmo processo. No caso da saída o display apresentado será o dia e hora de entrada e em baixo o dia, hora atual (de saída) e o tempo acumulado (tempo incrementado enquanto no interior) e dando reconhecimento ao sistema que este utilizador já se encontra no exterior.

De referir que em qualquer ponto deste processo poderá ser iniciado a rotina de manutenção tendo assim a prioridade sobre tudo.

Para cada ação diferente foi criando um método nesta classe, um para a apresentação inicial, leitura da identificação e password, validação do utilizador, realização da rotina de entrada, abertura e fecho da porta, formatação de hora e data de acordo com o pretendido e método que atualiza a estrutura de dados na qual os utilizadores estão guardados para fim de mudar o tempo acumulado e para as funções de criação e remoção de utilizadores da manutenção.

4 Conclusões

A partir da implementação destes módulos ficou-se a conhecer profundamente a relação software-hardware que possibilita a aprendizagem da programação de um objeto comum e útil á sociedade dando assim uma introdução a um dos objetivos do curso bem como da consequente profissão que é o estudo, conceção e implementação de um módulo.

Durante a realização deste projeto foram abertas novas visões de trabalho bem como aperfeiçoamento dos conteúdos de programação já lecionados tanto do plano do hardware bem como de software.
De tantas novas visões de trabalho apreendidas realça se a organização e o isolamento de partes como as mais importantes. A organização permitiu ter um apercebimento correto do que se passa sempre que algo ocorre sendo erro ou algo que fosse inesperado e também com a organização e calendarização mental permitiu uma maior eficiência no aproveitamento do tempo.

O isolamento de partes permite a localização de um erro num projeto que tem várias, mas que se testarmos isoladamente as partes saberemos mais rapidamente e com maior precisão onde será o erro e daí deriva também outro conceito que será a abstração e hierarquia pois permite que mesmo sejam mudados alguns módulos não será preciso a mudança de todas as partes envolvidas, só aquelas que estão em contacto direto.

As linguagens Java e CUPL foram cruciais para a programação de todos os componentes. O material necessário para montar todo o projeto foi: Breadboard, PALV750c, ATB, um registo implementado com Flip-Flops D, um teclado matricial 4*4 e por fim o Kit UsbPort.

A. Descrição CUPL do bloco *Key Decode*

```
/* *************** INPUT PINS *********************/
PIN    1     =  Mclk;
PIN [5..8]   = [L0..3];
PIN    9     =  ACK;

/* *************** OUTPUT PINS *********************/
PIN 14 = Wreg;
PIN [20..22] = [C0..2];
PIN [16..19] = [Q0..3];
PIN    23    =  DVAL;
PIN 15 = KPRESS;

/***************** PINNODES ************************/

PINNODE [29,30] = [X0,X1];

[Q0..3].AR = 'b'0;
[Q0..3].SP = 'b'0;
[Q0..3].CK = !Mclk;

CE = KSCAN;
Q0.t = CE;
Q1.t = Q0 & CE;
Q2.t = Q1 & Q0 & CE;
Q3.t = Q2 & Q1 & Q0 & CE;

S0 = Q2;
S1 = Q3;
S2 = Q0;
S3 = Q1;

/* ***************** Decoder************************/

C0 = S0 # S1;
C1 = !S0 # S1;
C2 = S0 # !S1;

/* ***************** MUX ***************************/

KPRESS = !(!S2&!S3&L0 # S2&!S3&L1 # !S2&S3&L2 # S2&S3&L3);


/* ***************** ALT NOMENCLATURA *****************/

DAV=KVAL;
KACK=DAC;

/* ***************** ASM ***************************/
[X0,X1].AR = 'b'0;
[X0,X1].SP = 'b'0;
[X0,X1].CK = Mclk;

Sequence [X0,X1]{
        Present 0
                out KSCAN;
                if !KPRESS next 0;
                if KPRESS next 1;
        Present 1
                out KVAL;
                if KACK next 2;
                default next 1;
        Present 2
                if !KACK & !KPRESS next 0;
                default next 2;
        }
```

B. Descrição CUPL do bloco *Key Buffer*

```
/* ***************** KEY BUFFER ***************************/

PINNODE [31,32] = [Y0,Y1];

[Y0,Y1].AR = 'b'0;
[Y0,Y1].SP = 'b'0;
[Y0,Y1].CK = Mclk;

sequence [Y0,Y1] {

        present 0
                if DAV next 1;
                default next 0;
        present 1
                out Wreg, DAC ;
                next 2;
        present 2
                out DVAL;
                if ACK next 3;
                default next 2;
        present 3
                if ACK next 3;
                default next 0;
```

C. Descrição CUPL do bloco *Serial Receiver*

```
Name     SerialReceiver ;
PartNo   00 ;
Date     27-Mar-18 ;
Revision 01 ;
Designer Engineer ;
Company  CCISEL ;
Assembly None ;
Location  ;
Device   v750c ;

/* *************** INPUT PINS *********************/
PIN  1  = SCLK                                    ;
PIN  2  = Mclk                                    ;
PIN  3  = SDX                                     ;
PIN  4  = accept                                  ;

/* *************** OUTPUT PINS ********************/
PIN  [18..23]   = [D0..D5]                        ;
PIN  17         = busy                            ;
PIN  16         = DXval                           ;
PIN  14         = init                            ;
PIN  15         = wr                              ;

/* *************** PINNODES ********************/
/*PINNODE[40,41]  = [dFlag,pFlag]                   ;*/
PINNODE[26..28] = [T0..T2]                        ;
PINNODE[29,30]  = [S0,S1]                         ;
PINNODE[31..33] = [C0..C2]                        ;
PINNODE[25] = [P0];

/* *************** Serial Receiver ********************/


/* *************** Shift Register ********************/
[D0..D5].SP='b'0;
[D0..D5].AR='b'0;
[D0..D5].CKMUX= SCLK;

SIN = SDX;

D0.d = (wr&SIN)#(!wr&D0);
D1.d = (wr&D0)#(!wr&D1);
D2.d = (wr&D1)#(!wr&D2);
D3.d = (wr&D2)#(!wr&D3);
D4.d = (wr&D3)#(!wr&D4);
D5.d = (wr&D4)#(!wr&D5);

/* *************** Counter ********************/

[T0..2].AR=init;
[T0..2].SP='b'0;
[T0..2].CKMUX=SCLK;

T0.t='b'1;
T1.t=T0;
T2.t=T0&T1;


pFlag=T2&T1&T0;
dFlag=T2&T1&!T0;


/* *************** Start ASM ********************/
[S0,S1].AR = 'b'0;
[S0,S1].SP = 'b'0;
[S0,S1].CK = Mclk;

Sequence [S0,S1]{
```

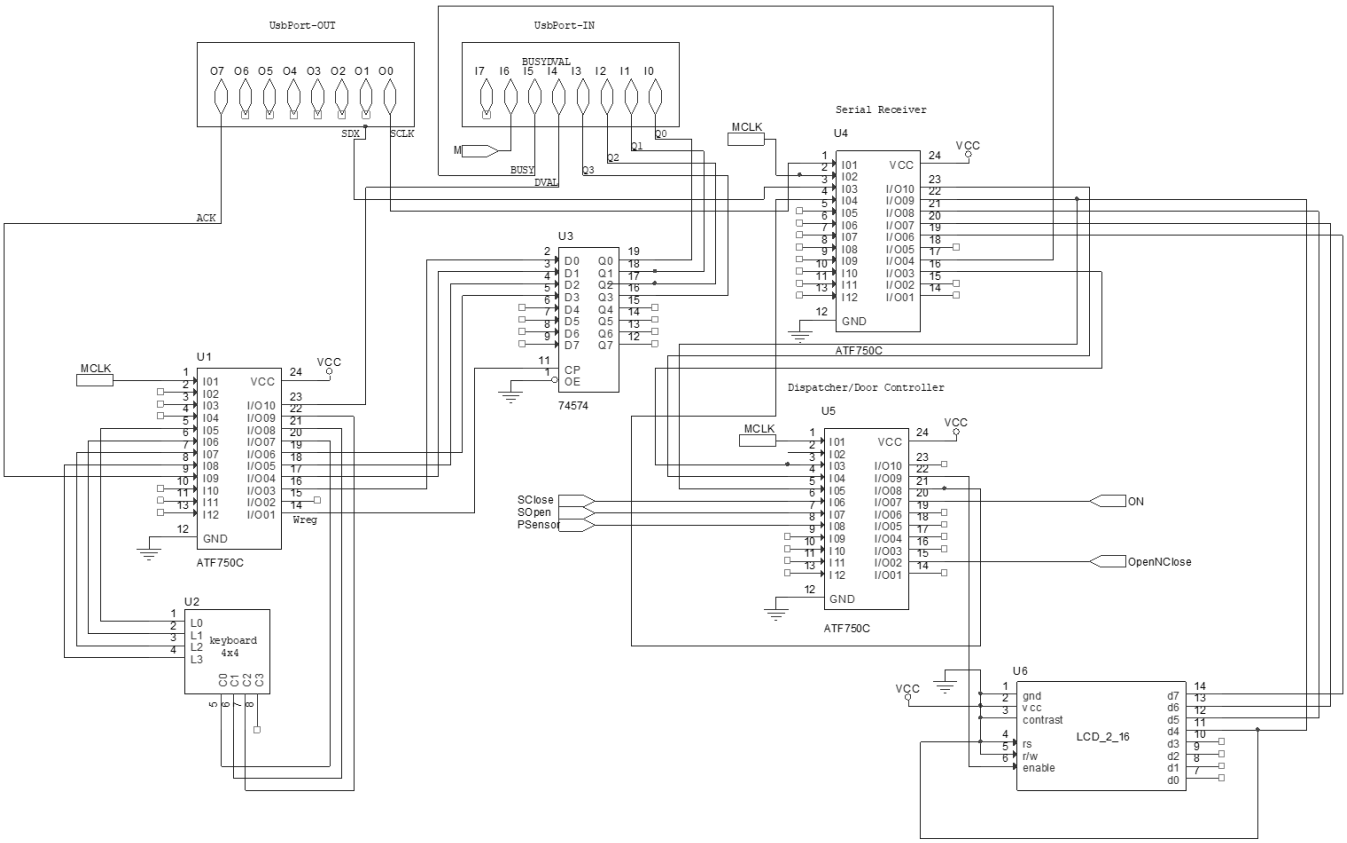
```
        Present 0
                if SCLK next 0;
                if !SCLK & !SDX next 0;
                if !SCLK & SDX next 1;
        Present 1
                if SCLK next 0;
                if !SCLK & SDX next 1;
                if !SCLK & !SDX next 2;

        Present 2
                out Start;
                default next 0;
        }


/* *************** Control ASM *********************/
RXerror = P0;


[C0..C2].AR = 'b'0;
[C0..C2].SP = 'b'0;
[C0..C2].CK = !Mclk;

Sequence [C0..C2]{
        Present 0
                out init;
                if Start next 1;
                default next 0;

        Present 1
                out wr;
                if !Start next 0;
                if dFlag next 2;
                default next 1;

        Present 2
                if !Start next 0;
                if pFlag & RXerror next 0;
                if pFlag & !RXerror next 3;
                default next 2;

        Present 3
                out busy,DXval;
                if accept next 4;
                default next 3;

        Present 4
                out busy;
                if !accept next 0;
                default next 4;

}

/* *************** Parity Check *********************/
P0.AR=init;
P0.SP='b'0;
P0.CK=SCLK;

P0.T= SDX;
```

D. Descrição CUPL do bloco *Dispatcher/DoorController*

```
Name     Dispatcher ;
PartNo   00 ;
Date     03-May-18 ;
Revision 01 ;
Designer G10 ;
Company  CCISEL ;
Assembly None ;
Location  ;
Device   v750c ;

/* *************** INPUT PINS *********************/
PIN  1   = Mclk                         ;
PIN  3   = Dval                         ;
PIN  4   = D5LnD                        ;
PIN  5   = D4OC                         ;
PIN  6   = Sclose                       ;
PIN  7   = Sopen                        ;
PIN  8   = Psensor                      ;




/* *************** OUTPUT PINS *********************/

PIN  22   =  WrL                        ;
PIN  21   =  done                       ;
PIN  20   =  On                         ;
PIN  15   =  OpenNClose                 ;


/* *************** PINNODES    *********************/
PINNODE[39,40]     =[Di0,Di1]                      ;
PINNODE[29..31]    =[Dc0..Dc2]                     ;

/* *************** Dispatcher    *********************/

[Di0..Di1].AR = 'b'0;
[Di0..Di1].SP = 'b'0;
[Di0..Di1].CK = Mclk;

Sequence [Di0..Di1] {
       Present 0
              if Dval & !D5LnD next 1;
              if Dval & D5LnD next 2;
              default next 0;

       Present 1
              out WrD;
              if finish next 3;
              default next 1;

       Present 2
              out WrL;
              default next 3;

       Present 3
              out done;
              if Dval next 3;
              default next 0;

}
```

```
/* *************** Door Controller *********************/
wr=WrD;
finish=doneDoor;

[Dc0,Dc1].AR = 'b'0;
[Dc0,Dc1].SP = 'b'0;
[Dc0,Dc1].CK = !Mclk;

Sequence [Dc0,Dc1]{
       Present 0
              if !wr next 0;
              if wr & D4OC next 2;
              if wr & !D4OC next 1;

       Present 1
              out On;
              if Sclose & !D4OC next 3;
              if !Sclose & Psensor next 2;
              if !Sclose & !Psensor next 1;

       Present 2
              out On,OpenNClose;
              if Sopen & D4OC next 3;
              if Sopen & !D4OC next 1;
              default next 2;

       Present 3
              out doneDoor;
              if wr next 3;
              default next 0;


}
```

E. Esquema elétrico



F. Código Java da classe *HAL*

```
import isel.leic.UsbPort;


public class HAL {
   private static  int lastout;
    public static void init(){
         UsbPort.out(~lastout);
         }
         public static boolean isBit(int mask){
            return (mask &in())!=0;

        }
        public static int readBits(int mask){
            return in()& mask;
            }

        public static void writeBits (int mask , int value){
           lastout=((~mask)&lastout)|(mask&value);
           out(lastout);


        }

        public static void setBits(int mask){
            lastout = lastout|mask;
            UsbPort.out(~lastout);
            }
            private static int in (){
                return ~UsbPort.in();

            }
            private static void out (int out){UsbPort.out(~out);}


        public static void clrBits(int mask){
            lastout=lastout&(~mask);
            UsbPort.out(~lastout);


        }

}
```

G. Código Java da classe *KBD*

```
import isel.leic.UsbPort;
import isel.leic.utils.Time;

public class KBD {


    public static final char NONE = 0;
   // private static char [] chars =
{'1','4','7','*','2','5','8','0','3','6','9','#','A','B','C','D'};
    private static char [] charsSimul =
{'1','2','3','4','5','6','7','8','9','*','0','#'};
    private static char lastchar;
    private static final int KEY = 15; //input mask : 0000 1111
    private static final int D_VAL = 16; //input mask :0001 0000
    private static final int ACK = 128; //output mask : 1000 0000

    public static void init(){


    }
    public static char getKey(){
        if(HAL.isBit(D_VAL)){
            int keycurrent= HAL.readBits(KEY);
            HAL.setBits(ACK);
            while (HAL.isBit(D_VAL)) ;
            HAL.clrBits(ACK);
            return charsSimul[keycurrent];
        }
        return NONE;

        }



    public static char waitKey(long timeout){
        timeout += Time.getTimeInMillis();
        while(Time.getTimeInMillis() < timeout){
            char k = getKey();
            if(k != NONE) return k;
        }
        return NONE;
    }

        }
```

H. Código Java da classe LCD

```
import isel.leic.UsbPort;
import isel.leic.utils.Time;

public class LCD {
        private static final int LINES = 2, COLS = 16;

    private static final int RS_MASK = 0x4;
    private static final int EN_MASK = 0x8;
    private static final int NIBBLE_MASK = 0xF0;

    private static void writeNibble(boolean rs, int data){

        if(!SerialEmitter.isBusy()) {
            data <<= 1;
            if (rs) data |= 1;
            SerialEmitter.send(SerialEmitter.Destination.LCD, data);
            Time.sleep(10);
        }

        }


        private static void writeByte(boolean rs, int data){
            int aux = data;
            aux>>=4;
            writeNibble(rs,aux);
            Time.sleep(5);
            writeNibble(rs,data);
            Time.sleep(1);
        }

        private static void writeCMD(int data){
            writeByte(false,data);
        }

        private static void writeDATA(int data){
            writeByte(true,data);

        }


        public static void init(){

            Time.sleep(15);
            writeNibble(false,0x03);
            Time.sleep(5);
            writeNibble(false,0x03);
            Time.sleep(1);
            writeNibble(false,0x03);
            Time.sleep(1);
            writeNibble(false,0x02);
            writeCMD(0x028);
            writeCMD(0x08);
            writeCMD(0x01);
            writeCMD(0x06);
            writeCMD(0x0f);
```

```
        }

        public static void write (char c){
            writeDATA(c);
        }

        public static void write (String txt){
            for(int i=0; i<txt.length(); i++) writeDATA(txt.charAt(i));
        }

        public static void cursor (int lin , int col){
            int DB7 = 0x80, DB6 = 0x40;
            if(lin>0) writeCMD(DB7+DB6+col);
            else writeCMD(DB7+col);

        }

        public static void clear (){
            writeCMD(0x01);

        }


}
```

I. Código Java da classe SerialEmitter

```
import isel.leic.utils.Time;

public class SerialEmitter {

    public enum Destination {DoorMechanism,LCD};
    private static final int SCLK = 0x01; //output
    private static final int SDX = 0x02; //output
    private static final int BUSY_MASK = 0x20;



    public static void start(){
        HAL.clrBits(SCLK);
        HAL.setBits(SDX);
        HAL.clrBits(SDX);
    }

    public static void send(Destination addr, int data) {
        start();
        int paridade = 0;
        int auxiliar = data;

        if(addr.ordinal() == 1){
            HAL.setBits(SDX);
            paridade++;
        }
        else HAL.clrBits(SDX);

        HAL.setBits(SCLK);
        Time.sleep(1);

        for(int i=0; i<5; i++){

            Time.sleep(1);
            if((auxiliar & 1) == 1) {
                HAL.setBits(SDX);
                paridade++;

            }
            else HAL.clrBits(SDX);
            HAL.clrBits(SCLK);
            auxiliar >>= 1;
            HAL.setBits(SCLK);
            Time.sleep(1);

            }

        if(paridade % 2 != 0)  HAL.setBits(SDX);
        else HAL.clrBits(SDX);

        HAL.clrBits(SCLK);
        Time.sleep(1);

        HAL.setBits(SCLK);
        Time.sleep(1);
```

```
    }
    public static boolean isBusy(){
        if (HAL.readBits(BUSY_MASK) == 1 ) return true;
        return false;
    }
}
```

J. Código Java da classe Door

```
public class Door {
    private static boolean end;

    public static void init(){
        end=false;

    }

    public static void open (int speed){
        speed <<= 1;
        speed |= 1;
        SerialEmitter.send(SerialEmitter.Destination.DoorMechanism,speed);
        if(isFinished()) end=true;

    }

    public static void close (int speed){
        speed <<=1;
        SerialEmitter.send(SerialEmitter.Destination.DoorMechanism,speed);
        if(isFinished()) end =true;
    }

    public static boolean isFinished(){
        return (!SerialEmitter.isBusy());


    }


}
```

K. Código Java da classe TUI

```
import isel.leic.utils.Time;

import java.time.LocalDate;
import java.time.LocalDateTime;

public class TUI {
    public static int readInteger(long timeout){
        char c =KBD.waitKey(timeout);
        return c-'0';

    }
    public static void writeLCD(String txt){
        LCD.write(txt);

    }
    public static void writeIntLcd(int i){
        LCD.write(""+i);

    }

    public static String time (){
       String time = "";

          if (LocalDateTime.now().getMinute() <= 9) {
              time = "" + LocalDate.now().getDayOfMonth() + "/" +
LocalDate.now().getMonthValue() + "/" + LocalDate.now().getYear() + " " +
LocalDateTime.now().getHour() + ":" + "0" + LocalDateTime.now().getMinute();
          }
        else
              time = "" + LocalDate.now().getDayOfMonth() + "/" +
LocalDate.now().getMonthValue() + "/" + LocalDate.now().getYear() + " " +
LocalDateTime.now().getHour() + ":" + LocalDateTime.now().getMinute();
          return time;

   }
    public static void init(){
        LCD.cursor(0,0);
        LCD.write(TUI.time());


    }

    public static void setCursor (int l, int c ){
        LCD.cursor(l,c);
    }

    public static void openDoor(){
        Door.open(15);
    }

    public static void closeDoor(){
        Door.close(15);
    }
    public static void clearLCD(){
        LCD.clear();
    }
```

L. Código Java da classe File Access

```
import isel.leic.utils.Time;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.Scanner;


public class FileAccess {

   public static String File = "USERS.txt";

    public static void load () throws FileNotFoundException {
        Time.sleep(10);
       try{
           BufferedReader rd = new BufferedReader(new FileReader(File));
           Scanner in = new Scanner(rd);

           for(;in.hasNextLine();) {
            String txt = in.nextLine();
            String[] splited = txt.split(";");

            Users.User user =null;
            if(splited.length==5){
                user =new
Users.User(Integer.parseInt(splited[0]),Integer.parseInt(splited[1]),splited[2],Integer.parseI
nt(splited[3]),Integer.parseInt(splited[4]));            }
            else{
                user =new
Users.User(Integer.parseInt(splited[0]),Integer.parseInt(splited[1]),splited[2],Integer.parseI
nt(splited[3]));
            }


            Users.userlist.add(user);

        }
        }
        catch(Exception e){
            System.out.println("File not found");
        }



    }
}
```

Código Java da classe Users

```
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.ArrayList;

public class Users {

    public static ArrayList<User> userlist  = new ArrayList ();

    public static void loadUsers ()throws FileNotFoundException {
        FileAccess.load();
    }
```

```
public static String searchUser(int id){
    for (User m : userlist){
        if(m.getNumber()==id) return m.getName();
    }
    return null;
}

public static int getPassword(int id){
    for (User m : userlist){
        if(m.getNumber()==id) return m.getPass();
    }
    return -1;
}
public static String getName(int id){
    for (User m : userlist){
        if(m.getNumber()==id) return m.getName();
    }
    return null;
}
public static User getUser(int id){
    for (User m : userlist){
        if(m.getNumber()==id) return m;
    }
    return null;
}

public static class User {

    private int identification;
    private int pass;
    private String worker;
    private int timespent;
    private boolean InNOut;
    private long entryHour;
    private String entryDay;


    public User (int ID,int password, String name, int acumulatedTime){
        pass=password;
        worker=name;
        identification=ID;
        timespent=acumulatedTime;
        InNOut = true;
    }

    public User (String name, int password){
        pass=password;
      //  identification=getLowerID();
        InNOut = true;
        timespent=0;
    }

    public int getNumber() {
        return identification;
    }

    public String getName() {
        return worker;
    }
    public int getPass(){
```

```
        return pass;
    }
    public int getTime(){
        return timespent;

    }

    public void setInNOut(boolean inNOut) {
        InNOut = inNOut;
    }

    public boolean getInNOut(){
        return InNOut;
    }
    public int getID(){
        return identification;
    }

    public void setEntryHour(long entryHour) {
        this.entryHour = entryHour;
    }

    public long getEntryHour() {
        return entryHour;
    }

    public void addTime(long time){
        timespent+=time;
    }

    public void saveParameters(PrintWriter pw) {
        pw.print(identification+";"+pass+";"+worker+";"+timespent);
    }

    public String getEntryDay() {
        return entryDay;
    }

    public void setEntryDay(String entryDay) {
        this.entryDay = entryDay;
    }
}
```

M. Código Java da classe M

```
public class M {

    private static final int MSignal = 0x80; // input mask

    public static boolean MVerification(){
        if(HAL.isBit(MSignal)) return true;
        return false;
    }
}
```

N. Código Java da classe Maintenance

```
import java.util.Scanner;

public class Maintenance {

    public static void commands() {

        Scanner in = new Scanner(System.in);
        TUI.clearLCD();
        TUI.setCursor(0, 1);
        TUI.writeLCD("Out of Service");
        TUI.setCursor(1, 5);
        TUI.writeLCD("Wait");
        System.out.println("Turn M key to off, to terminate the maintenance mode.");
        System.out.println("Commands: NEW, DEL, LST, or OFF");


        while (M.MVerification()) {
            String str = in.nextLine();
            System.out.println("Maintenance> ");


            if (str.equals("OFF")) {
                System.exit(0);
            }


            if (str.equals("NEW")) {
                System.out.println("User ? ");
                String name = in.next();
                System.out.println("pass ?");
                int pass = in.nextInt();
                Users.User user = new Users.User(App.generateID(), pass, name, 0);
                Users.userlist.add(user);
                System.out.print("Adding user :" + user.getID() + ":" + user.getName());
                App.updateUserList();

            }

            if (str.equals("DEL")) {
                System.out.println("UIN?");
                int id = in.nextInt();
                System.out.println("Remove user" + id + ":" + Users.getName(id));
                System.out.println("Y/N?");
                if (in.next().equals("Y")) {
                    System.out.println("User " + id + ":" + Users.getUser(id) + "removed.");
                    Users.userlist.remove(Users.getUser(id));
                    App.updateUserList();

                } else {
                    System.out.println("Command aborted.");
                }
            }

            if (str.equals("LST")) {
                for (Users.User m : Users.userlist) {
                    if(!m.getInNOut()) System.out.println(m.getID() + "," + m.getName()+"," +
m.getEntryHour());

                }
            }

        }
```

```
        }
    }
```

O. Código Java da classe Log

```
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.PrintWriter;
import java.time.LocalDate;
import java.time.LocalTime;

public class Log {

    public static String LOGFile = "LOG.txt";
    public static void Logged (Users.User u) {

        try {
            FileOutputStream f = new FileOutputStream(LOGFile, true);
            PrintWriter pr = new PrintWriter(f);
            LocalDate ld = LocalDate.now();
            LocalTime lt = LocalTime.now();

            pr.write(ld + " " + lt + " " + (u.getInNOut() ? "->" : "<-") + " " + u.getID() +
":" + u.getName());
            pr.println();
            pr.close();
        }
        catch(Exception e){ System.out.println("FIle not found");}

    }



}
```

P. Código Java da classe App

```
import isel.leic.utils.Time;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.time.DayOfWeek;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.util.concurrent.TimeUnit;

public class App {
    private static Users.User updatedUser;

    public static void main(String[] args) throws FileNotFoundException {
        HAL.init();
        SerialEmitter.init();
        LCD.init();
        TUI.clearLCD();
        Users.loadUsers();
        for (; ; ) {
            TUI.clearLCD();
            TUI.init();
            if (M.MVerification()) Maintenance.commands();
            if (userInput()) {
                routine();
            }

        }
    }

    private static void routine() throws FileNotFoundException {

        TUI.clearLCD();
        TUI.setCursor(0, 5);
        TUI.writeLCD("HELLO");
        TUI.setCursor(1, 3);
        TUI.writeLCD(updatedUser.getName());
        Time.sleep(2000);
        TUI.clearLCD();
        showEntry();
    }

    public static int readID(long timeout) {

        TUI.setCursor(1, 0);
        LCD.write("UIN:??? ");
        TUI.setCursor(1, 4);

        int c = 4;
        int count = 0;
        int aux = 0;

        timeout += Time.getTimeInMillis();



        do {
            int k = TUI.readInteger(timeout);

            if (k >= 0 && k <= 9) {
                TUI.setCursor(1, c++);
                TUI.writeLCD(k + "");
                aux += k;
```

```
                aux *= 10;
                ++count;

            }
            if (k == -6) {
                aux = 0;
                count = 0;
                TUI.setCursor(1, 4);
                TUI.writeLCD("??? ");
                c = 4;
                TUI.setCursor(1, c);
            }
            if (M.MVerification()){
                Maintenance.commands();
                return -1;
            }

        } while ((Time.getTimeInMillis() < timeout) && count < 3);

        if (count == 3) return aux / 10;

        else return -1;

    }

    public static int readPass(long timeout) {
        TUI.setCursor(1, 0);
        LCD.write("PIN:????");

        TUI.setCursor(1, 4);


        int c = 4;
        int count = 0;
        int aux = 0;
        timeout += Time.getTimeInMillis();

        do {
            int k = TUI.readInteger(timeout);
            if (k >= 0 && k <= 9) {
                TUI.setCursor(1, c++);
                TUI.writeLCD(  "*");
                aux += k;
                aux *= 10;
                ++count;
            }
            if (k == -6) {
                aux = 0;
                count = 0;
                TUI.setCursor(1, 4);
                TUI.writeLCD("???? ");
                c = 4;
                TUI.setCursor(1, c);
            }


        } while ((Time.getTimeInMillis() < timeout) && count < 4);


        if (count == 4) return aux / 10;
        else return -1;
    }

    public static boolean userInput() {


        int a = readID(15000);
```

```
        if(Users.getUser(a)==null)return false;
        int b = readPass(15000);

        int c = 0;


        if (Users.getUser(a) != null) {
            updatedUser = Users.getUser(a);
            c = Users.getPassword(a);
            return c == b;
        }

        return false;


    }

    public static String convertDays(LocalDate day) {
        LocalDateTime d = LocalDateTime.now();
        String dia = getFormattedWeekDay(d.getDayOfWeek());
        return dia;

    }

    public static String convertEntryDay(String day) {
        String d = updatedUser.getEntryDay();
        String daay = "";
        switch (d) {
            case "MONDAY":
                daay = "seg.";
                break;
            case "TUESDAY":
                daay = "ter.";
                break;

            case "WEDNESDAY":
                daay = "qua.";
                break;

            case "THURSDAY":
                daay = "qui.";
                break;

            case "FRIDAY":
                daay = "sex.";
                break;

            case "SATURDAY":

                daay = "sab.";
                break;

            case "SUNDAY":
                daay = "dom.";
                break;

        }
        return daay;
    }

    public static String getFormattedWeekDay(DayOfWeek d) {
        String str = "";
        switch (d) {
            case MONDAY:
                str = "seg.";
                break;
            case TUESDAY:
```

```
                str = "ter.";
                break;

            case WEDNESDAY:
                str = "qua.";
                break;

            case THURSDAY:
                str = "qui.";
                break;

            case FRIDAY:
                str = "sex.";
                break;

            case SATURDAY:
                str = "sab.";
                break;

            case SUNDAY:
                str = "dom.";
                break;
        }
        return str;
    }


    public static void showEntry() throws FileNotFoundException {

        if (updatedUser.getInNOut()) {
            updatedUser.setEntryHour(Time.getTimeInMillis());
            updatedUser.setEntryDay(LocalDate.now().getDayOfWeek() + "");
            Log.Logged(updatedUser);
            updatedUser.setInNOut(false);


            TUI.setCursor(0, 0);
            TUI.writeLCD(convertDays(LocalDate.now()) + " " + LocalTime.now().getHour() + ":"
+ (LocalTime.now().getMinute() < 10 ? "0" + LocalTime.now().getMinute() :
LocalTime.now().getMinute()));
            Time.sleep(3000);
            TUI.clearLCD();
            TUI.setCursor(0, 2);
            TUI.writeLCD(updatedUser.getName());
        } else {
            updatedUser.addTime(Time.getTimeInMillis() - updatedUser.getEntryHour());
            Log.Logged(updatedUser);
            updatedUser.setInNOut(true);
            updateUserList();

            TUI.setCursor(0, 0);
            long minutoEntrada = (int)(updatedUser.getEntryHour() / ((1000 * 60)) % 60);
//TimeUnit.MILLISECONDS.toHours(updatedUser.getEntryHour());
            long horaEntrada = (int)(updatedUser.getEntryHour() / ((1000 * 60 * 60)) % 24);
//TimeUnit.MILLISECONDS.toMinutes(updatedUser.getEntryHour());
            int minutos = (int) TimeUnit.MILLISECONDS.toMinutes(updatedUser.getTime());
            int horas = (int) TimeUnit.MILLISECONDS.toHours(updatedUser.getTime());

            TUI.writeLCD(convertEntryDay(updatedUser.getEntryDay() + "") + "" +
(horaEntrada+1) + ":" + (minutoEntrada < 10 ? "0" + minutoEntrada : minutoEntrada));
            TUI.setCursor(1, 0);
            TUI.writeLCD(convertDays(LocalDate.now()) + LocalTime.now().getHour() + ":" +
LocalTime.now().getMinute() + " " + horas + ":" + (minutos < 10 ? "0" + minutos : minutos));
            Time.sleep(3000);
            TUI.clearLCD();
            TUI.setCursor(0, 2);
            TUI.writeLCD(updatedUser.getName());
```

```
            }
            openingDoor();
            Time.sleep(2000);
            closingDoor();
            Time.sleep(2000);
            TUI.clearLCD();
            updateUserList();
        }

        public static void openingDoor() throws FileNotFoundException {


            TUI.setCursor(1, 2);
            TUI.writeLCD("Door Opened");
            do Door.open(5);
            while (!Door.isFinished());
            Time.sleep(1000);
            TUI.clearLCD();

        }

        public static void closingDoor() {

            TUI.setCursor(1, 0);
            TUI.writeLCD("Closing Door..");
            do Door.close(5);
            while (!Door.isFinished());
            Time.sleep(1000);
            TUI.clearLCD();
            Time.sleep(1000);


        }

        public static void updateUserList() {
            try {
                PrintWriter pw = new PrintWriter(new File("USERS.TXT"));
                for (Users.User u : Users.userlist) {
                    u.saveParameters(pw);
                    pw.println();

                }
                pw.close();

            } catch (FileNotFoundException e) {
                e.printStackTrace();
            }
        }

        public static int generateID() {
            int id =(int) (Math.random()*(999-1)+1);

            for (Users.User u :Users.userlist) {
                if(id==u.getID())id =(int) (Math.random()*(999-1)+1);

            }
              return id;
        }
    }
```

Q. Código Java da classe Users

```
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.ArrayList;

public class Users {

    public static ArrayList<User> userlist  = new ArrayList ();

    public static void loadUsers ()throws FileNotFoundException {
        FileAccess.load();
    }

    public static String searchUser(int id){
        for (User m : userlist){
            if(m.getNumber()==id) return m.getName();
        }
        return null;
    }

    public static int getPassword(int id){
        for (User m : userlist){
            if(m.getNumber()==id) return m.getPass();
        }
        return -1;
    }
    public static String getName(int id){
        for (User m : userlist){
            if(m.getNumber()==id) return m.getName();
        }
        return null;
    }
    public static User getUser(int id){
        for (User m : userlist){
            if(m.getNumber()==id) return m;
        }
        return null;
    }

    public static class User {

        private int identification;
        private int pass;
        private String worker;
        private int timespent;
        private boolean InNOut;
        private long entryHour;
        private String entryDay;


        public User (int ID,int password, String name, int acumulatedTime){
            pass=password;
            worker=name;
            identification=ID;
            timespent=acumulatedTime;
            InNOut = true;
        }

        public User (int ID,int password, String name, int acumulatedTime,int EntryHour){
            pass=password;
            worker=name;
            identification=ID;
            timespent=acumulatedTime;
            InNOut = false;
```

```
            entryHour=EntryHour;

        }

        public int getNumber() {
            return identification;
        }

        public String getName() {
            return worker;
        }
        public int getPass(){
            return pass;
        }
        public int getTime(){
            return timespent;

        }

        public void setInNOut(boolean inNOut) {
            InNOut = inNOut;
        }

        public boolean getInNOut(){
            return InNOut;
        }
        public int getID(){
            return identification;
        }

        public void setEntryHour(long entryHour) {
            this.entryHour = entryHour;
        }

        public long getEntryHour() {
            return entryHour;
        }

        public void addTime(long time){
            timespent+=time;
        }

        public void saveParameters(PrintWriter pw) {
            pw.print(identification+";"+pass+";"+worker+";"+timespent+(!getInNOut()? ";" +
entryHour :""));
        }

        public String getEntryDay() {
            return entryDay;
        }

        public void setEntryDay(String entryDay) {
            this.entryDay = entryDay;
        }
    }
}
```